AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) In a local server that receives data from one or more remote clients over a data transport protocol, a method of generating an initial sequence number (ISN) for use by a remote client when assigning sequence numbers to one or more data packets to be sent to the local server, the initial sequence number generated in a manner that prevents the local server from being attacked while maintaining reliable data transfer, the method comprising:

generating a random input key using a secret maintained by the local server;

receiving a connection identifier key that includes connection information for at least the remote client;

securely initializing a hash function with at least a portion of the random input key and at least a portion of the connection identifier key; ~~for determining an intermediate value;~~

calculating an intermediate value using the initialized hash function;

creating a monotonically increasing counter for ensuring that a same connection identifier does not have data collisions from competing sequence numbers within a predetermined period of time, and for ensuring randomness of the initial sequence number on a per connection basis for preventing attacks on the local server, the counter taking both timer information and connection rate information as input;

incrementing the counter a fixed value based on a passage of a predetermined time period;

detecting a connection rate for the local server, the connection rate being determined by detecting each connection to the local server and a connection rate evaluator determining the number of connections per unit time;

determining a variable amount based upon the connection rate, the variable amount being determined by at least comparing the connection rate to a threshold value and by choosing a random number between a minimum ISN increase per connection and a maximum ISN increase per connection;

incrementing the counter ~~[[a]]~~ by the variable amount; ~~depending upon the connection rate for local server, the increment being based upon the connection rate;~~

combining the fixed value and the variable amount to create a random value; and

combining the intermediate value and the random value using a monotonically increasing mathematical function to generate the initial sequence number.

2. (Previously Presented) The method of claim 1, wherein if the connection rate is below the threshold value, the fixed value is further incremented based on each connection established with the local server.

3. (Original) The method of claim 2, wherein based on the fixed value, if a remote client's data transfer rate while connected to the local server is less than a specified byte rate then the connection identifier used by the remote client the is allowed immediate re-connection to the local server after the remote client disconnects.

4. (Original) The method of claim 1, wherein the connection identifier key further includes connection information for one or more of the local server port, local server routing address, remote port and remote routing address.

5. (Previously Presented) The method of claim 4, wherein the data transport protocol is Transmission Control Protocol (TCP), and wherein the local and remote routing addresses are Internet Protocol (IP) addresses.

6. (Original) The method of claim 1, wherein at least a second connection is made between the local server and a second remote client, and wherein the method further including the acts of:

receiving a second connection identifier key that includes connection information for at least the second remote client;

securely initializing the hash function with at least a portion of the random input key and at least a portion of the second connection identifier key for determining a second intermediate value of a second initial sequence number;

based on at least a portion of the second connection identifier key, creating a second monotonically increasing counter for ensuring that a same connection identifier does not have

data collisions from competing sequence numbers within a predetermined period of time, and for ensuring randomness of the second initial sequence number on a per connection basis for preventing attacks on the local server;

incrementing the second counter the fixed value based on the passage of the predetermined time period;

incrementing the second counter a second variable amount depending upon a rate of connections with the local server and for those connections associated with the second counter, wherein if the rate of connections with the local server and for those connections associated with the second counter is beyond a threshold value the variable increment is based on an elapsed time, otherwise the variable increment is based on each connection established with the local server and associated with the second counter; and

combining the second intermediate value, the fixed value and the second variable amount for generating the second initial sequence number.

7. (Original) The method of claim wherein 1, wherein the arbitrary information maintained as a secret by the local server is based on timing, state conditions for the local server, or both, at boot up time of the local server, which include one or more of a time of day, a day of month, a month, a year, a local server hard drive head position, and whether input was detected by hardware of the local server.

8. (Cancelled)

9 (Previously Presented) The method of claim 1, wherein if the connection rate is beyond the threshold value the variable increments up to an amount of 0x000022FB every millisecond, otherwise the variable increment is an amount between 16 K and 32 K.

10. (Original) The method of claim 1, wherein the monotonically increasing counter is shared by at least two connections at the same time.

11 – 20 (Cancelled)

21. (Previously Presented) For a local server that receives data from one or more remote clients over a data transport protocol, a computer program product comprising computer readable storage media carrying computer executable instructions that implement a method of generating an initial sequence number for use by a remote client when assigning sequence numbers to one or more data packets to be sent to the local server, the initial sequence number generated in a manner that prevents the local server from being attacked while maintaining reliable data transfer, the method comprising the method of claim 1.

22. (Previously Presented) The computer program product of claim 21, wherein if the connection rate is below the threshold value, the fixed value is further incremented based on each connection established with the local server.

23. (Original) The computer program product of claim 22, wherein the fixed value is 25.6 K, and wherein if a remote client's data transfer rate while connected to the local server is less than 256 K then the connection identifier used by the remote client the is allowed immediate re-connection to the local server after the remote client disconnects.

24. (Original) The computer program product of claim 21, wherein the connection identifier key further includes connection information for one or more of the local server port, local server routing address, remote port and remote routing address.

25. (Previously Presented) The computer program product of claim 24, wherein the data transport protocol is Transmission Control Protocol (TCP), and wherein the local and remote routing addresses are Internet Protocol (IP) addresses.

26. (Original) The computer program product of claim 21, wherein at least a second connection is made between the local server and a second remote client, and wherein the computer executable instructions further implement the method including the acts of:

receiving a second connection identifier key that includes connection information for at least the second remote client;

securely initializing the hash function with at least a portion of the random input key and at least a portion of the second connection identifier key for determining a second intermediate value of a second initial sequence number;

based on at least a portion of the second connection identifier key, creating a second monotonically increasing counter for ensuring that a same connection identifier does not have data collisions from competing sequence numbers within a predetermined period of time, and for ensuring randomness of the second initial sequence number on a per connection basis for preventing attacks on the local server;

incrementing the second counter the fixed value based on the passage of the predetermined time period;

incrementing the second counter a second variable amount depending upon a rate of connections with the local server and for those connections associated with the second counter, wherein if the rate of connections with the local server and for those connections associated with the second counter is beyond a threshold value the variable increment is based on an elapsed time, otherwise the variable increment is based on each connection established with the local server and associated with the second counter; and

combining the second intermediate value, the fixed value and the second variable amount for generating the second initial sequence number.

27. (Original) The computer program product of claim wherein 21, wherein the arbitrary information maintained as a secret by the local server is based on timing, state conditions for the local server, or both, at boot up time of the local server, which include one or more of a time of day, a day of month, a month, a year, the local server hard drive position, and whether input was detected by hardware of the local server.

28. (Cancelled)

29. (Previously Presented) The computer program product of claim 21, wherein when the connection rate is beyond the threshold value the variable increments up to an amount of 0x000022FB every millisecond, otherwise the variable increment is an amount between 16 K and 32 K.

30. (Original) The computer program product of claim 21, wherein the monotonically increasing counter is shared by at least two connections at the same time.

31 – 40. (Cancelled)